L Number	Hits	Search Text	DB	Time stamp
1	722	(expert near2 system) same (rule$1 or (rule near2 based) or rule-set) same (template$1 or table$1 or form$1)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:19
2	203	((expert near2 system) same (rule$1 or (rule near2 based) or rule-set) same (template$1 or table$1 or form$1)) and (tree or trie or hierarchial)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:19
3	40	(((expert near2 system) same (rule$1 or (rule near2 based) or rule-set) same (template$1 or table$1 or form$1)) and (tree or trie or hierarchial)) and (control$3 near3 (table$1 or form$1 or template$1))	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:22
4	2	((((expert near2 system) same (rule$1 or (rule near2 based) or rule-set) same (template$1 or table$1 or form$1)) and (tree or trie or hierarchial)) and (control$3 near3 (table$1 or form$1 or template$1))) and 707/$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:25
5	2	DAG and RAK	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:27
6	2999	((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:41
7	385	(((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based)) and (GUI or "graphical user interface")	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:32
8	3	((((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based)) and (GUI or "graphical user interface")) and (DAG or "dynamic application generator")	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:31
9	223	(((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based)) and (GUI or "graphical user interface") and (web or web-site or web-based or browser)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:34
10	52	((((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based)) and (GUI or "graphical user interface") and (web or web-site or web-based or browser)) and 707/$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:35

11	10	((((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based)) and (GUI or "graphical user interface") and (web or web-site or web-based or browser)) and "expert system" and 707/$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:35
12	187	(((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based)) and (control near2 table$1)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:41
13	31	((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based) same (GUI or "dynamic user interface")	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:55
14	16	(((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based) same (GUI or "dynamic user interface")) and 707/$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:46
15	49	((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based) same (GUI or "graphical user interface")	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:46
16	20	(((build$3 or generat$3) near5 (template$1 or table$1 or form$1)) same (rule$1 or (rule near2 set) or rule-based) same (GUI or "graphical user interface")) and 707/$.ccls.	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 14:03
17	3	((expert near2 system) same (rule$1 or (rule near2 based) or rule-set) same (template$1 or table$1 or form$1)) and DAG	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB	2004/05/03 13:55

US-PAT-NO: 5815717

DOCUMENT-IDENTIFIER: US 5815717 A

TITLE: Application program and documentation generator system and method

---------- KWIC ---------

Brief Summary Text - BSTX (12):

In order to increase the available performance of program generators, various forms of **expert** systems have been incorporated. Such **expert** systems include finite state automata, forward and backward chaining **rule** based inference engines, fuzzy logic inference systems and context sensitive or guided editor systems. Although the use of such **expert** systems can greatly increase the apparent functionality of a program generator system, such **expert** systems have typically been applied in an application processing environment that, based on predefined **rules,** limits the perceived complexity of the operative system to the particular data input and requests made of the **expert** system. As such, these **expert** systems present an environment within which an application appears to have been constructed. Compilable or interpretable application program code is not generated. Consequently, such environments are difficult to design and maintain and make little provision for the production of documentation that is specific to a particular working application program.

Detailed Description Text - DETX (4):

As the application structures and sequences are created or subsequently modified, the resulting structures and sequences are parsed through a substantially hierarchical **expert** system that successively applies program **rules** 18, skill **rules** 20 and syntax **rules** 22 that culminate in the generation of a discrete module or set of modules of compilable or interpretable code 24, independent documentation, or both code 24 and corresponding documentation 26. The program **rules** 18 are provided in a **rule** base organized to provide a set of common **rules** and discrete sets of **rules** specific to particular program types. Although program types may be arbitrarily defined consistent with the present invention, the presently preferred set of program types include data collection, data processing, and data output program types. Additional program types, such as one providing for the definition of an **expert** system that can be used separately or in support of the other program types can also be defined. Further, program sub-types may be associated as detailed categorizations of the program **rules** 18. Consistent with a preferred embodiment of the present invention that is oriented toward accounting or accounting type application generation, the program sub-types preferably include maintenance, transactional, and detailed analysis screens subordinate to the data collection program type, a posting process subordinate to the processing program type and columnar report, inquiry report, and form reports subordinate to the output program type. Thus, as the program **rules** 18 are applied to the application structure and sequences 12, appropriate sets and sub-sets of the program **rules** are data dependently identified and executed toward the realization of the

application program specifically corresponding to the particular application structure and sequences 12 to which the program **rules** 18 are applied. Consequently, the present invention provides most directly for the presence and utilization of potentially conflicting program **rules** that may appropriately exist in support of different types and sub-types of application programs. Since some **rules** are not associated with any particular program type or sub-type, some of the program **rules** 18 are generically and therefore efficiently defined for use in generation of all types of application programs.

Detailed Description Text - DETX (29):

Once the application structure 36 and application sequences 38 for a program have been constructed through the operation of the image editor 32, an application author 44 is invoked to generate a corresponding set of compilable or interpretable code. The application author operates from the application structure 36, application sequences 38, the contents of the help file 40 and a corresponding set of programs, skill and syntax **rules** as stored by the knowledge base 34. The application author 44, operating generally as an **expert** system, applies program, skill and syntax **rules** to the available definition of the application structure 36 in concert with the specific application sequences 38 to produce the code for a program of the particular type and sub-type being generated. That is, a program of the selected program type and sub-type is constructed by the operation of the **expert** system of the author 44 based on the defined application structure 36 and available applications sequences 38. In accordance with the present invention, the code produced by the author 44 is the product of the program **rules** that define a particular program type and sub-type to the extent that such **rules** are consistent with the provided application structure 36 and application sequences 38. Since the code statements are effectively independently generated by the operation of the author 44 based on the application structure and sequences 36, 38 the individual code statements are inherently unified with respect to the number, naming and typing of all variables instantiated by the author 44. No further unification of variables or mediation of the interfaces between segments of a single program is required. Interfaces between segments, to the extent that they are required, are automatically matched by the operation of the **expert** system of the author 44 based on the definition of the application structure 36 and the application sequences that define data items and fields that are shared by different segments of a common program.

Detailed Description Text - DETX (30):

Finally, a documentation publisher 46 effectively operates in parallel with the application author 44. The **expert** system implementing the documentation publisher 46 operates based on the application structure and sequences 36, 38 and **rules** and information provided from the knowledge base 34 and help file 40. In the preferred embodiment of the present invention, the publisher **expert** system 46 is simplified by relying upon the application author 44 to first effectively process the application sequences 38 to particularly provide detailed help text describing process elements implemented by the functional logic defined by the sequences. For example, the help text for a field defined as a three digit positive number might be: "The value of the Accounting Period field cannot exceed 999, but may be as small as 0." Consequently, the detailed help file text is generated by the application author 44 on behalf of the documentation publisher 46. This information thus closely represents the

particular code generated by the application author and alleviates the requirement of the documentation publisher 46 might otherwise have to completely reconstruct the code through reparsing the application structure and sequences 36, 38. Alternately, the **expert** system underlying the documentation publisher 46 could be merged with the application author 44 so that fully formed documentation is directly generated by the author 44. Equivalently, the documentation publisher 46 may duplicate the **expert** system of the author 44 and operate directly from the information sources utilized by the author 44. In any event, the documentation publisher 46 operates on the information provided to the publisher 46 to generate text documents that correspond closely to the code produced by the application author 44. In particular, this includes a merge of the explanatory text entered into the help file 40, the screens, processes, and report formats designated by the operation of the publisher 46 itself, the structure and menu organization of the documented program as determined from the application structure 36 including any definitions and explanations processed by the publisher 46 based on documentation structure definition **rules** obtained from the knowledge base 34.

Detailed Description Text - DETX (33):

In the preferred embodiment of the present invention, the programs 52 are partitioned into relatively discrete functional operations. For example, within an accounts payable topic, a particular program 52 may isolate and perform the function of receiving and storing input defining an invoice. Another program 52 may be defined as responsible for posting invoice records to general ledger accounts. Another program may be specified to select and print a reconciliation report. Thus, the logical partitioning of the programs 52 can be discretely identified by program type and sub-type thereby supporting the operation of the **expert** system underlying the author 44 in generating the corresponding program code. Further, additional program types and sub-types may be readily added or defined and the set of program **rules** modified or extended to appropriately support new program types and sub-types.

Detailed Description Text - DETX (48):

Finally, the various definitions generated by the image editor are provided preferably as each definition is completed to the sequence generator 66. The definitions are evaluated by the sequence generator 66 to generate sets of application sequences. The definition evaluation is preferably performed by an **expert** system underlying the sequence generator 66. As feature packets and functional logic are evaluated by this **expert** system, reference is made to a **control table** 64 that, in combination with **rule** table 62, operates as an abstract, multi-level data look-up facility utilized in support of generating definite application sequences. The **control table** identifies sets of **rule** table entries while the **rule** table provides the abstracted behavioral characteristics for each basis referenced in the feature packets. A simple basis may have a single **rule** table entry that is properly used in all instances where the basis may appear. A more complex basis, or function, defined as a basis that may be subject to some ambiguity in its intended use, may be first evaluated against the **control table** to determine a particular set of basis **rules** to consider in further evaluating the complex basis or function. Thus, for example, a function reference to a field name in the context of a data input process may imply the need for establishing the input focus transversal of the field relative to others. However, the same function in the context of

a report process would imply a significantly different functionality, particularly one related to the optimum retrieval order of data for this and related fields. Functions, representing complex basis, are referenced through the **control table** 64. These functions are summarized in Table IV.

Detailed Description Text - DETX (63):

Referring now to FIG. 6, a detail 90 of the sequence generator 66 and application sequences produced by the sequence generator 66 are show. As previously established, the sequence generator 66 operates from the application structure 36, presented as file definitions 92 and program definitions 94, and the field definitions 96 produced by the image editor 32. These definitional inputs are processed by the **expert** system of the sequence generator 66 directly or indirectly through reliance on the **control table** 64 to identify and select behavioral characteristics of a specific basis from the **rule** table 62 as identified by attributes of the field definitions 96. In the preferred embodiment of the present invention, the sequence generator 66 operates to generate discrete sequences that are functionally descriptive of particular aspects of the input definitions. Specifically, program sequences 98 are generated to define the apparent ordered execution of the process represented by a particular segment 54. For example, program sequences will define the order that screen display fields are visited. Program sequences will also implicitly define the order of retrieving, editing, and saving record data.

Detailed Description Text - DETX (73):

Frames 112 of the application sequences are generally sequentially evaluated by the **expert** system underlying the application author 44. As indicated in FIG. 7, programmer specifications, representing the collected application sequences are received as an input to the application author in combination with the application structure 36 as represented by the file definitions 92 and program definitions 94. In addition, the application author 44 has access to the knowledge base 34 for retrieval of program **rules** 120, test **rules** 122, basis **rules** 124, skill **rules** 126, and syntax **rules** 128. Based on the program type and sub-type obtained effectively from the program definition 94, the applicable set of program **rules** 120 and corresponding test **rules** 122 are selected for evaluation by the author 44 against the programmer specifications. These selected program **rules** anticipate and impose a general framework to the functions and relationships presented by the programmer specifications. As a consequence, the program **rules** 120 serve to select out of the programmer specifications the information from the available application sequences needed to satisfy the program **rules** 120. The test **rules** 122 serve as high level qualifications tests to determine whether different subsets of the program **rules** 120 are to be utilized in view of the particular programmer specifications being supplied to the application author 44. Consequently, the comprehensive structure and flow of a particular code module generated by the application author 44 is inferentially dependant on the fields and field relationships initially established based on user input.

Detailed Description Text - DETX (77):

Referring now to FIG. 8, the documentation publisher 46 operates generally in parallel with the application author 44, though with the notable distinction of the dependency upon the help file 40 as being prior augmented by the author

44. The publisher 46 depends on the application structure 36 as provided by the file definitions 92 and program definitions 94, which include the segment and record definitions, as well as the programmer specifications provided from the sequence generator 66. Specifically, the document publisher 46 implements a relatively simple **expert** system that evaluates document **rules** 130, obtained from the knowledge base 34, primarily against the information provided by the help file 40. The document **rules** 130 serve to guide the construction of the documentation from the help file information based generally on the application structure 36. In particular, the menu lists associated with the current topic of the program definitions 94 is utilized to organize the body of the documentation. The programmer specifications are evaluated, again generally on a frame by frame basis, to construct screen and report format representations suitable as illustrations within the documentation. Key words, such as function key names and basis types that occur in the help file 40 are used to trigger the evaluation of reserved word **rules** 132. These reserved word **rules** 132 are provided preferably to support expanded explanations of key concepts, operational features, or methods of use in a non-repetitive manner. Reserved word **rules** 132 may also support key word indexing and creation of glossary entries for key words.

Detailed Description Text - DETX (79):

Thus, a complete application generator system and method, providing direct support for the design, implementation, maintenance and documentation of essentially custom application programs has been described. The system requires a minimum amount of information to be input by a user in order to specify the functionality of the resulting application program. The information input is largely constrained to the definition of the application structure and logical attributes of fields that specify the characteristics of field and the relationships between fields that inferentially define the functions necessary to carry out the realization of the attribute defined functions and relationships. From these attributes and the application structure, an over-specification of application sequences is created that fully describe the inferred detailed relationships and qualifications of the attributes defined by user input. This over-specification is then reduced by an **expert** system that selects and utilizes needed and applicable portions of the application sequences to realize the control logic necessary to construct an application program consistent with the program types specified as part of the application structure. A complete application program is thereby constructed through reduction by syntax **rules** to a specific programming language. A parallel reduction of the prompted and automatically collected textual information also provides for the generation of application program documentation highly correlated to the specific application produced.

Detailed Description Text - DETX (80):

Various modifications and alternate implementations of the present invention are contemplated and may be readily resorted to by those of skill in the art without departing from the nature and scope of the present invention. In particular, many different implementations of the **expert** systems described above may be utilized in any particular embodiment. Direct **rule** parsing engines as well as backward chaining inference engines may be readily utilized in realizing the **expert** systems of the present invention. In addition, the **expert** systems of the author and publisher may be implemented as a single

expert system or chained sequentially to provide for the production of code and documentation. Accordingly, the present invention may be practiced otherwise than as described above though within the scope of the present invention particularly as defined by the appended claims.

Detailed Description Paragraph Table - DETL (6):

TABLE IV Control Table Functions

ALT KEY Defines how program elements are to be selected by an alternate index key reference in different contexts. AUTONUMBER Defines how ordered program elements are to be sequenced in different contexts. ELEMENT Defines the process for creating a program element in different contexts. ENTRY Defines traversal order for fields in different contexts. LOCATION Determines how the location of a field or program element is to be determined in different contexts. MAIN KEY Defines how program elements are to be selected by the main index key reference in different contexts. ORDER Determines the method of defining the order of program records or elements in different contexts. RECORD TYPE Determines the type of a program record in different contexts. UIP FIELDS Determines the set of additional fields or program elements that must be defined in support of the further rule processing of a program segment in different contexts.

US-PAT-NO: 6535883

DOCUMENT-IDENTIFIER: US 6535883 B1

TITLE: System and method for creating validation rules used to confirm input data

---------- KWIC ---------

Detailed Description Text - DETX (9):

Now that the components of the personal computer 10 and mobile computer 30 have been described, the validation **rules** program 15 will be discussed in further detail. The validation **rules** program 15 of the present invention generates an easy-to-use **graphical user interface (GUI)** to allow a service provider employee to quickly create a set of validation **rules** for a form, without assistance from an MPA developer. To do this, the validation **rules** program 15 represents and displays a form and its fields as a tree structure. The tree structure for a particular **form is generated from a user-selected form** template which is a file containing the form name and the names for the used fields for the form. The form name is displayed as the root node of the tree, and the names of the fields are displayed as leaf nodes. The **GUI** of the validation **rules** program includes a menuing interface for adding validation **rules** to fields, and for selecting expressions for implementing tests in validation **rules**. Commonly-used expressions are predefined as expression templates in the validation **rules** program 15, and are presented in a menu for quick selection by the user. An expression template includes an operator and blanks for the user to fill out for the one or more operands. The various expression templates are described in plain English, or another natural language.

Current US Original Classification - CCOR (1):
707/100

Current US Cross Reference Classification - CCXR (3):
707/102

Current US Cross Reference Classification - CCXR (4):
707/201

US-PAT-NO: 6629098

DOCUMENT-IDENTIFIER: US 6629098 B2

TITLE: Method and system for validating data submitted to a database application

---------- KWIC ---------

Brief Summary Text - BSTX (4):

In data collection systems wherein data is solicited and input via an entry data entry form made up of multiple data entry fields, it is typical that when the data entry form is submitted to the database application for entry, each data entry field is validated against a predetermined criteria. This process is generally referred to as validating the data input. This validation process is often used to ensure that data is input in the proper format and within a reasonable range of expected values. For example, a data entry field which solicits a telephone number should not typically include letters. Thus, during the validation process, the criteria established for this data entry field which requires that all data input into the telephone number field be numeric will be checked. To assure validation consistency among all applications using a database, the validation criteria is typically defined by a set of validation rules. The validation rules are often included as a part of, and enforced by, the database application. If the criteria is not met, validation will fail and will require correction of the entered data before it can be accepted and stored in the database. Typically, once the submitted data has failed to be validated, no further data entry fields on the submitted data entry form will be reviewed regardless of whether or not the other data fields contain valid data which meets all predetermined criteria for validation. This failure in data validation results in the database application generating an exception indicating that the submitted data entry form contains data which does not meet the predefined criteria for validation. The data entry form is presented to a user again. The user can then make corrections to data entry fields as may be necessary and then resubmit the data entry form. Upon re-submission, the data entry fields of the data entry form are again reviewed (submitted for validation). Where multiple errors have been made in the entry of data in the data fields of the submitted form, the typical database system will reject the submitted data form by generating multiple exceptions and representing the data form to a user for correction until each and every invalid data field meets the predetermined criteria. This is a time consuming, resource consuming, iterative process. Typical database systems cause the graphical user interface (GUI) used to present the data entry form to a user, to iterate and be presented to the user for correction each time validation of data entry fails. This increases the time and resources required to validate all fields of data entered.

Detailed Description Text - DETX (9):

With reference to FIG. 4 and FIG. 2 a diagram illustrating one embodiment of the system of the present invention is shown. There is an application 150,

which includes an application-programming interface (API) 152. Application 150 may be, for example, an e-commerce application or the like. Application 150 may be installed and run on client-side terminal 200 or downloaded from database server 250 as, for example, an applet. Application 150 causes a data entry form 602 to be generated, which incorporates a question or questions to be presented to a user as a GUI 600 (FIG. 2) on, for example, a display device 202 (FIG. 1). Once the user has completed answering the questions presented via entering data into data fields 61 through 64, application 150 causes a database (DB) update record 155 to be created. Database update record 155 specifies the data input by the user for a particular set of questions (51 through 54). This record is then passed through the API 152 to the server application 253, which in turn performs the database update via the database (DB) software 160. If this update fails because, for example, the data does not meet the criteria specified by the validation rules, the server application 253 will interact with the database software 160 to validate as many data fields as possible. Each error reported to the server application 253 will be added to an exception set as an exception. An exception specifies, for example, the data input field that contains data that cannot be validated. After all fields of data have been reviewed (validated), the exception set is returned to the application 150 which causes the data entry form 602 to be re-presented to the user for correction/re-entry of answer information to conform with the pre-determined criteria. If no exceptions are contained in the exception set, the server application 253 and the database software 160 causes the database 251 to be updated to include the data. When the application 150 receives an empty exception set, it continues with the next step (often the capture and submission of a different set of data to the database).

Current US Original Classification - CCOR (1):
707/6

Current US Cross Reference Classification - CCXR (1):
707/102

Current US Cross Reference Classification - CCXR (2):
707/2